

# Di[M]O: Distilling Masked Diffusion Models into One-step Generator

Yuanzhi Zhu, Xi Wang, Stéphane Lathuilière, Vicky Kalogeiton

*LIX, École Polytechnique, CNRS, IPP, Inria, Univ. Grenoble Alpes, CNRS, LJK*



# Content

---

- Preliminaries
- Methods
- Results

# Text

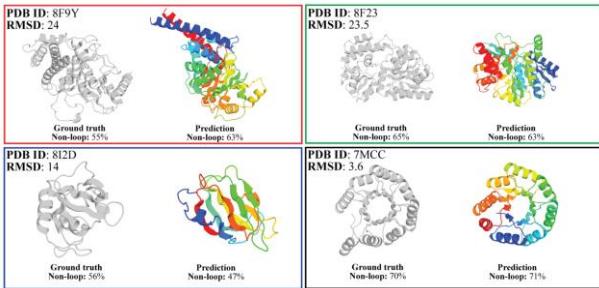
## Multi-round Dialogue Case

User	Could you please share with me what the initial two lines of the renowned poem 'The Road Not Taken'?
LLaDA	Certainly! The initial two lines of the renowned poem "The Road Not Taken" by Robert Frost are: "Two roads diverged in a yellow wood, And sorry I could not travel both." These lines set the stage for the poem, introducing the idea of a speaker facing a choice between two paths.
User	Please help me translate into Chinese.
LLaDA	Sure! The first two lines of "The Road Not Taken" by Robert Frost can be translated into Chinese as: "两条路分岔在黄色的树林中，遗憾我不能同时走"
User	Please translate into German.
LLaDA	Sure! The first two lines of "The Road Not Taken" by Robert Frost can be translated into German as: "Zwei Wege trennen sich im gelben Wald, und leider konnte ich nicht beide tun."
User	Please also write a poem about life choices. I require 5 sentences, and the first word of each sentence starts with C as the first letter.
LLaDA	Certainly! Here's a poem about life choices, with each sentence starting with the letter "C": Choices cascade through life's journey, Creating connections with each decision, Careful consideration leads to better paths, Courageous choices shape our destiny, Choosing wisely guides us through the years.

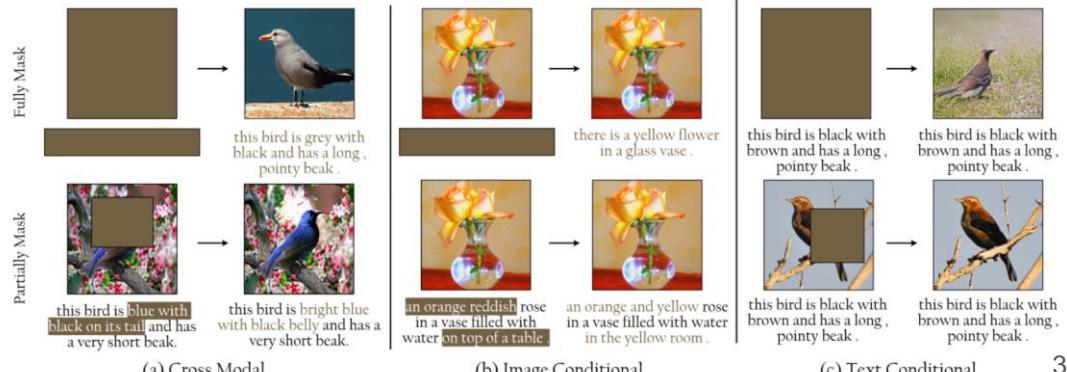
# Image



# Protein

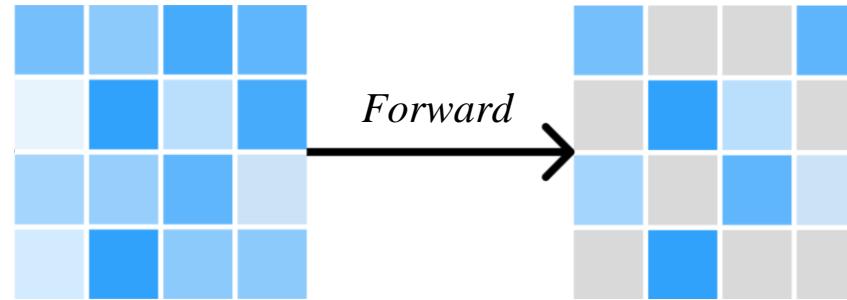


# Multi-modality



# Mask Diffusion Models

**Forward Process:** mask *independently* each token at position  $i$

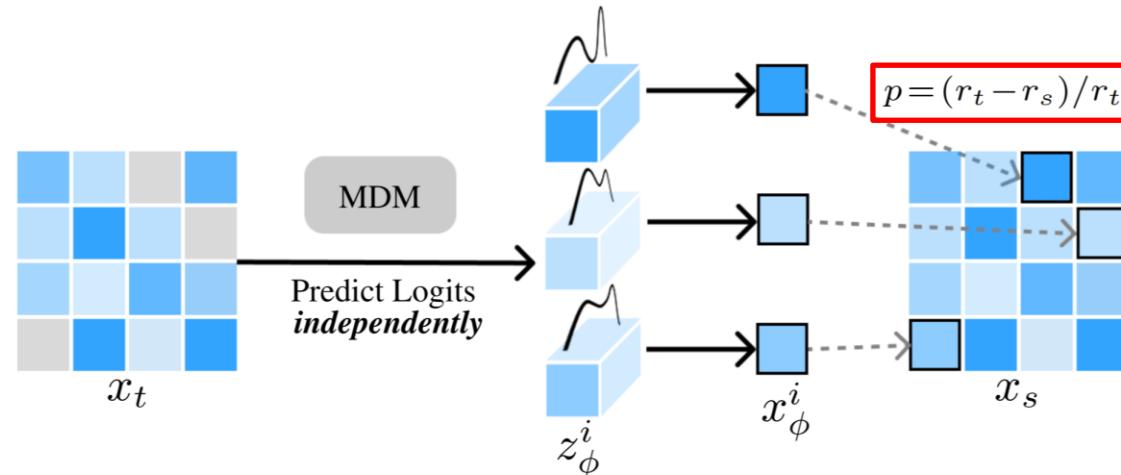


$$q_{t|0}(x_t|x_0) = \prod_{i=0}^{L-1} \text{Cat}(x_t^i; (1 - r_t)\delta(x_0^i) + r_t\delta([\mathbb{M}]))$$

# Mask Diffusion Models

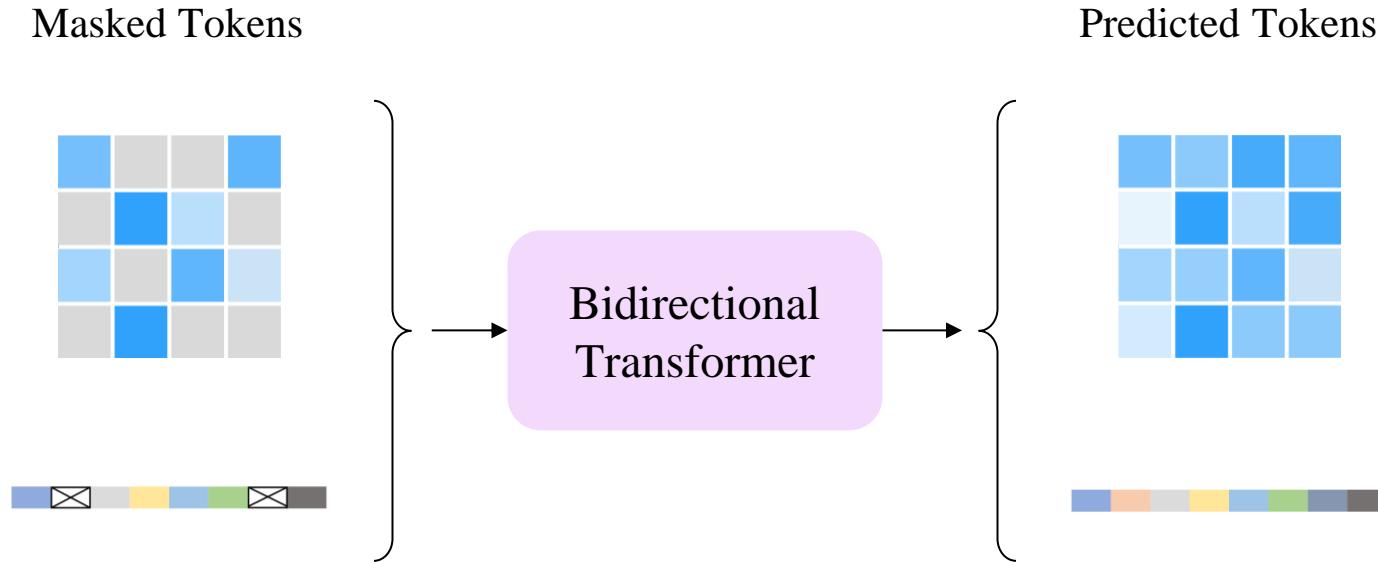
**Reverse Process:** fill masks

1. In parallel predict each token independently
2. Replace masks using strategies (e.g. **random**)



$$\text{Predicted Probability: } p_\phi(x_0^i | x_t) := \text{softmax} \left( z_\phi^i(x_t) / \tau \right)$$

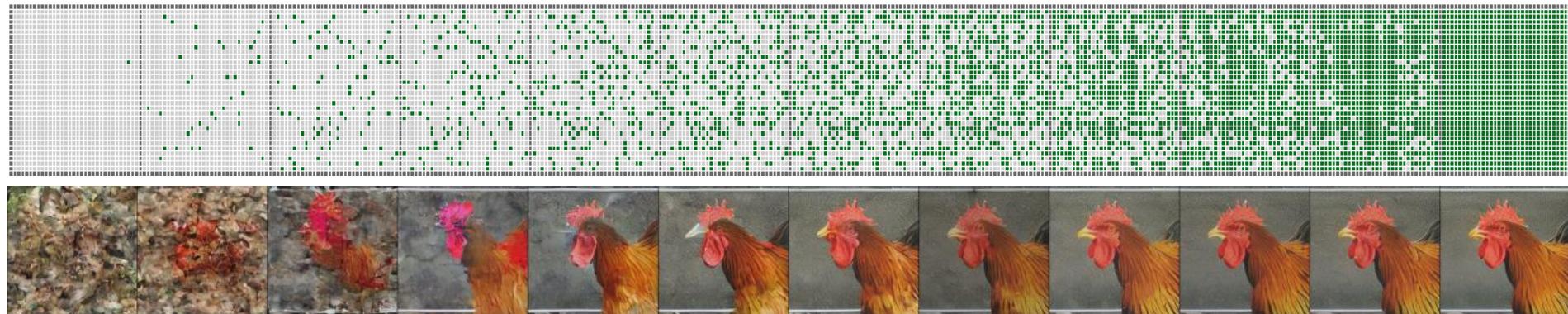
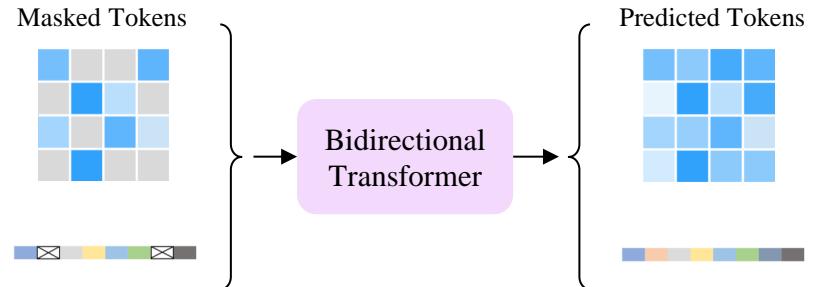
# Mask Diffusion Models



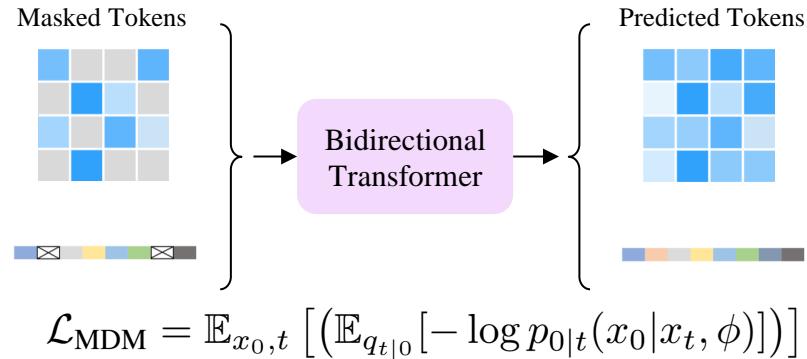
$$\mathcal{L}_{\text{MDM}} = \mathbb{E}_{x_0, t} \left[ (\mathbb{E}_{q_{t|0}}[-\log p_{0|t}(x_0|x_t, \phi)]) \right]$$

# Mask Diffusion Models

## Sampling Process



# Promise of MDM



**Theorem:** Minimizing the *expected* cross-entropy loss is equivalent to maximizing the variational lower bound of the log-likelihood, i.e.:

$$-\mathbb{E}_{p_{\text{data}}(x_0)} [\log p_{\theta}(x_0)] \leq \mathcal{L}_{\text{MDM}}(\theta)$$

**MDM loss** (maximize likelihood) + **Transformer** (less inductive bias) + **Data** → *Scalability* ?

# Limitations of MDM

- Incorrect joint distribution for multi-token prediction 😞
- Need sufficient steps for better performance 😞

## Ideas in Inference-time Scaling can Benefit Generative Pre-training Algorithms

Jiaming Song, Linqi Zhou

Luma AI

Multi-token prediction (MTP) is of great interest to the language modeling community because of its potential to achieve faster inference [GIR<sup>+</sup>24], which allows efficient inference-time scaling. However, the current multi-token prediction models often predict the softmax values of multiple tokens in parallel, which is a naïve conditional independence assumption (*i.e.*, naïve Bayes). We argue that this inference design greatly limits the capacity of the model distribution and more efforts should be spent resolving this fundamental issue.

## Theoretical Benefit and Limitation of Diffusion Language Model

Guhao Feng<sup>\*1</sup> Yihan Geng<sup>\*1</sup> Jian Guan<sup>2</sup> Wei Wu<sup>2</sup> Liwei Wang<sup>1</sup> Di He<sup>1</sup>

such as a reasoning chain—we show that the required sampling steps must scale linearly with sequence length to obtain “correct” sequences, thereby eliminating MDM’s efficiency advantage over autoregressive models. Our analysis estab-

[2503.07154] Ideas in Inference-time Scaling can Benefit Generative Pre-training Algorithms

[2502.09622] Theoretical Benefit and Limitation of Diffusion Language Model

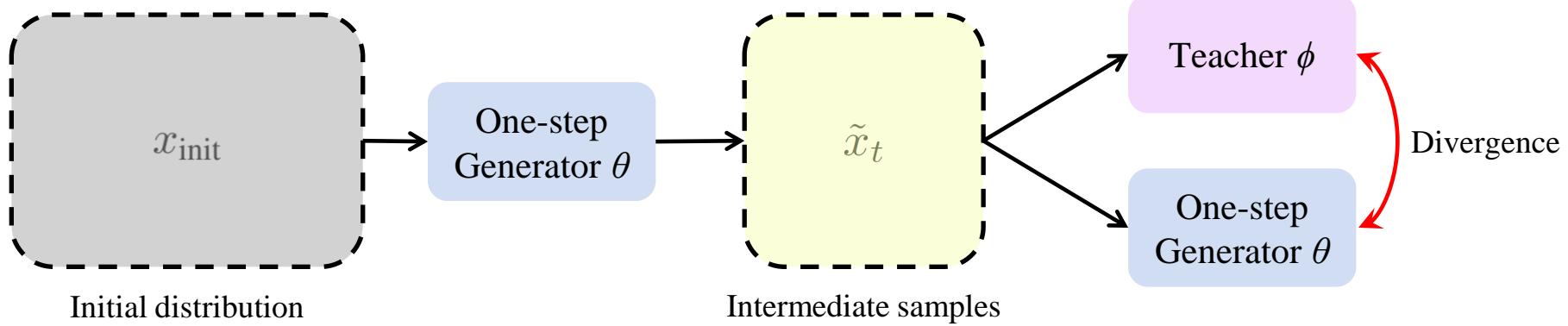
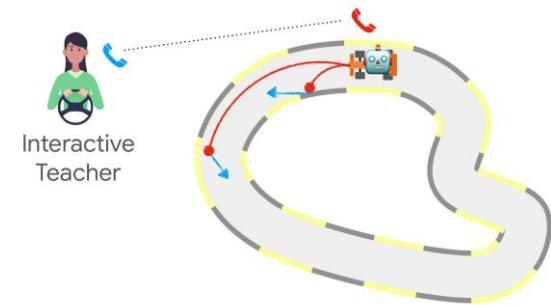
# Content

---

- Preliminaries
- Methods
- Results

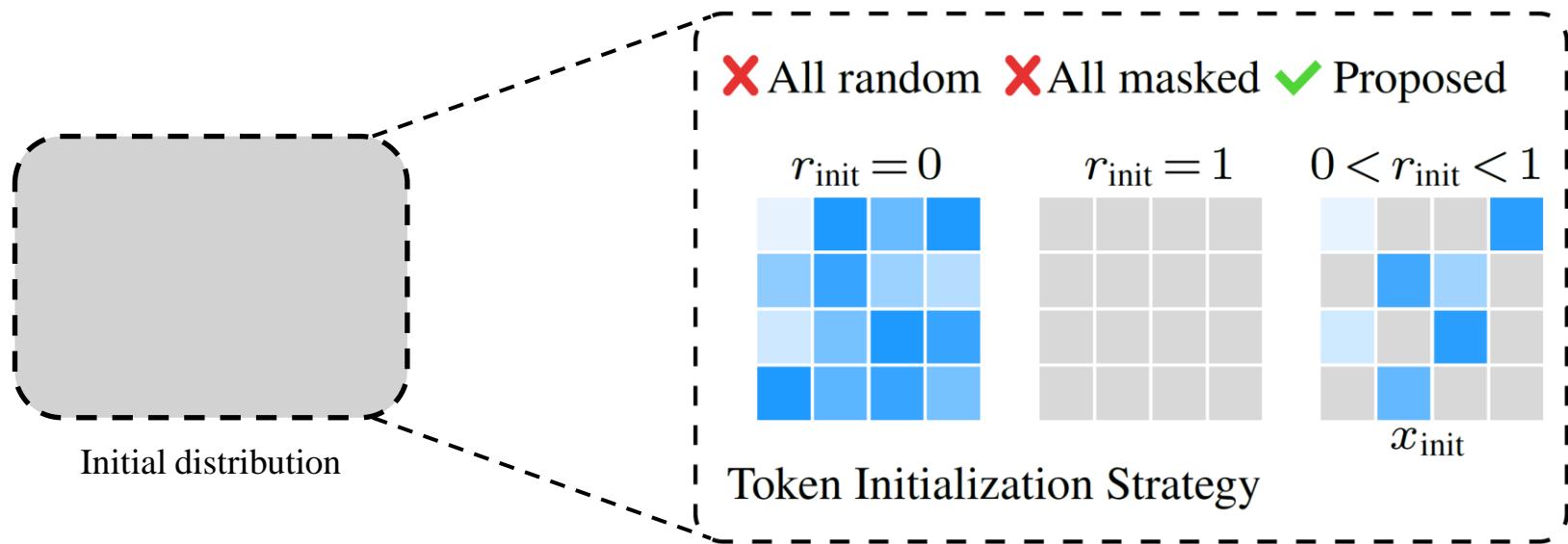
# Main Idea: Distillation in an *On-Policy* fashion

Match teacher and generator for all possible intermediate samples



$$\mathcal{L}_{\text{Di}[M]O}(\theta) := \mathbb{E}_{x_{\text{init}}, t} [w(t) (\mathbb{E}_{q_{t|0}} [D(p_\phi || p_\theta)(\tilde{x}_t)])]$$

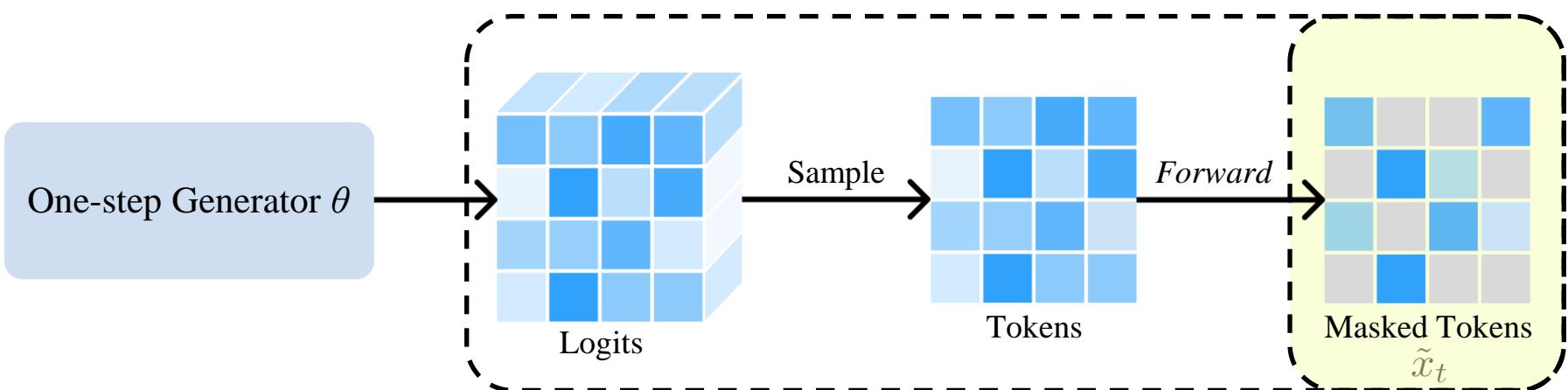
# Initial Distribution



Close to the input distribution (*masked token sequences*) used to train the teacher model

# Intermediate Samples

Via *forward mask diffusion process*



Consider loss on each intermediate state  $\tilde{x}_t$

$$\mathcal{L}_{\text{Di}[\mathbb{M}]O}(\theta) := \mathbb{E}_{x_{\text{init}}, t} [w(t) (\mathbb{E}_{q_{t|0}} [D(p_\phi || p_\theta)(\tilde{x}_t)])]$$

Further decompose into *Token-level* Divergence

$$D(p_\phi \| p_\theta)(\tilde{x}_t) := \frac{1}{L_M} \sum_{\substack{i=1 \\ \tilde{x}_t^i = [\mathbb{M}]}}^L D(p_\phi(x_0^i | \tilde{x}_t) \| p_\theta(x_0^i | \tilde{x}_t))$$

## Gradient of Divergence:

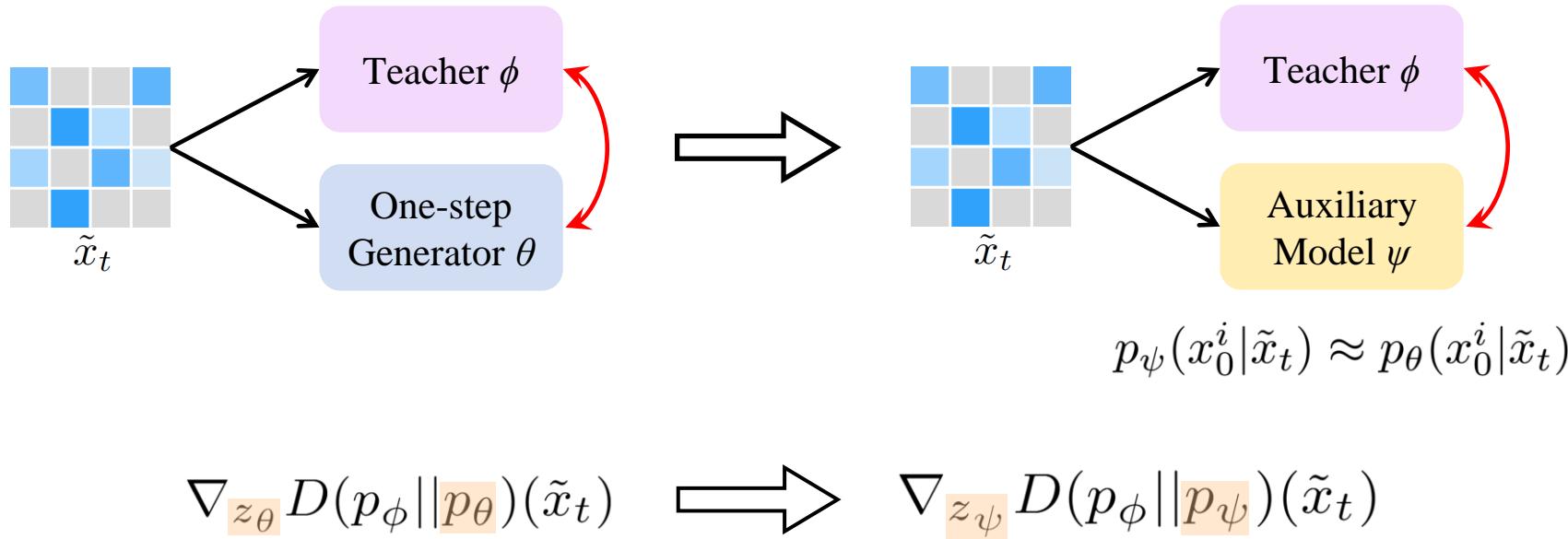
$$\nabla_{\theta} D(p_{\phi} || p_{\theta})(\tilde{x}_t) = \nabla_{z_{\theta}} D(p_{\phi} || p_{\theta})(\tilde{x}_t) \frac{dz_{\theta}(\tilde{x}_t)}{d\theta}$$

Intractable terms

$$p_{\theta}(x_0^i | \tilde{x}_t) = \text{softmax}(z_{\theta}(\tilde{x}_t))$$

Approximation:  $p_\theta(x_0^i | \tilde{x}_t)$

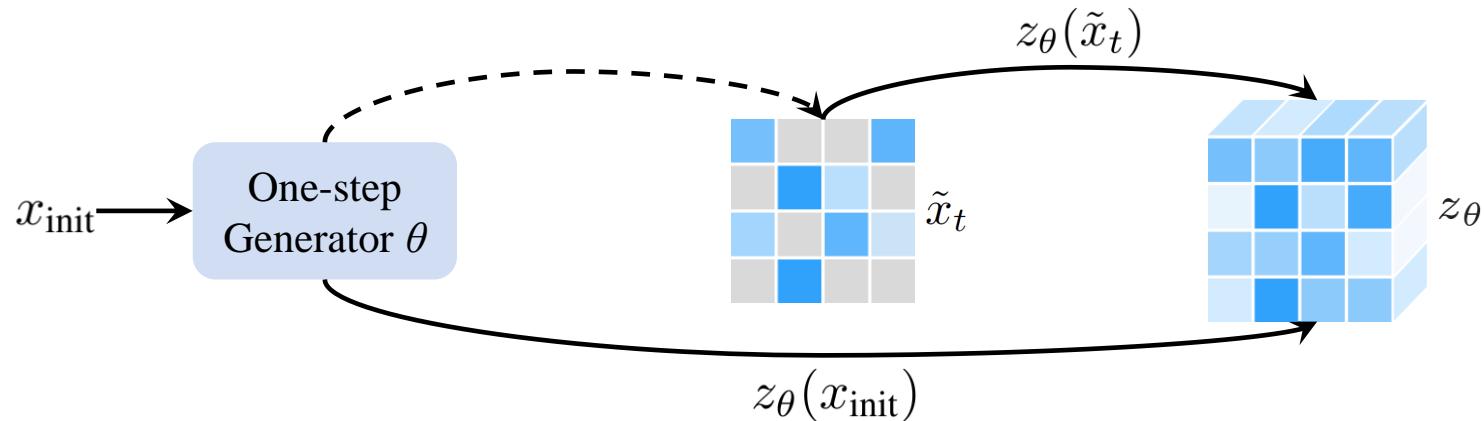
Introduce an **auxiliary model** to approximate the student output on intermediate samples



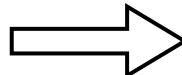
Approximation:  $z_\theta(\tilde{x}_t)$

We approximate the output logits based on the **consistency assumption**

Also, we require this term as effective gradient for the generator

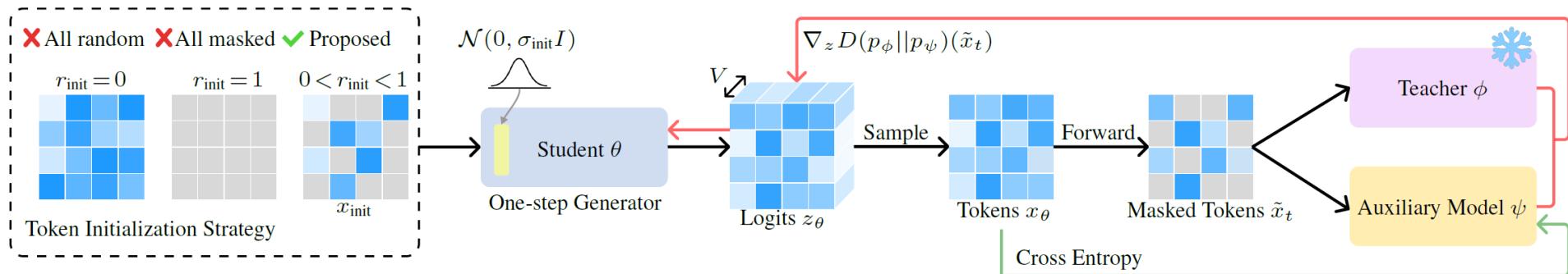


$$\frac{dz_\theta(\tilde{x}_t)}{d\theta}$$



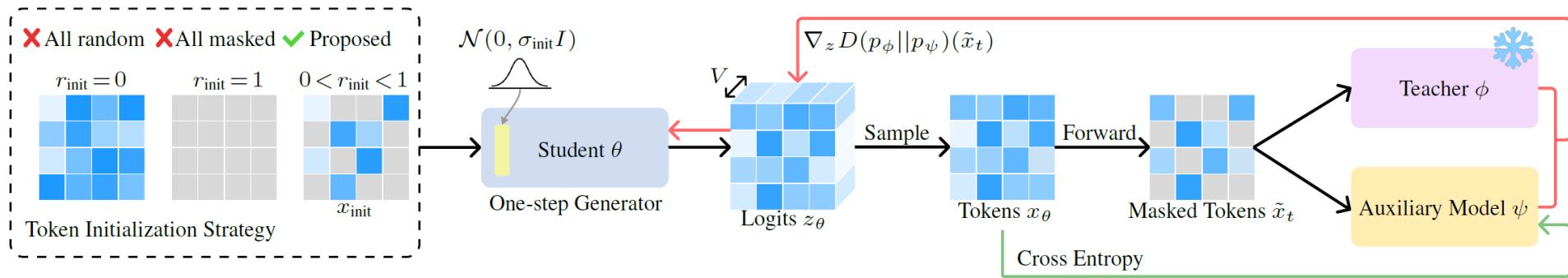
$$\frac{dz_\theta(x_{\text{init}})}{d\theta}$$

# Overview of the Method

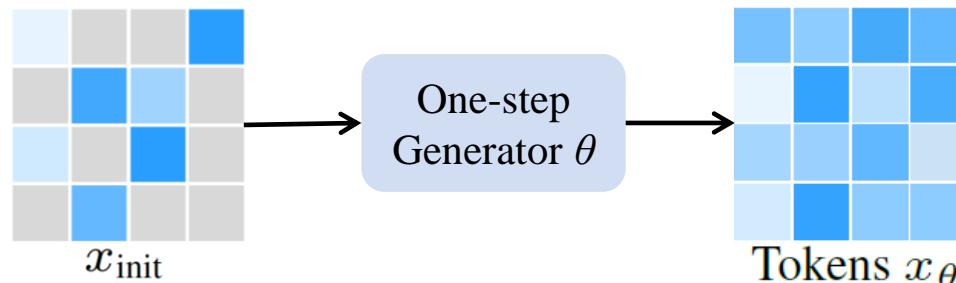


$$\nabla_\theta \mathcal{L}_{\text{Di}[M]O} \approx \mathbb{E}_{x_{\text{init}}, t} \left[ w(t) \left( \mathbb{E}_{q_{t|0}} \left[ \begin{array}{c} \nabla_{z_\psi} D(p_\phi || p_\psi)(\tilde{x}_t) \\ \frac{dz_\theta(x_{\text{init}})}{d\theta} \end{array} \right] \right) \right]$$

# Overview of the Method



- From random initial states to samples from *correct joint distribution* 😊
- Greatly reduce the sampling steps → One step generation 😊



# Overview of the Method

---

## Algorithm 1 Di[M]O Distillation

---

**Require:** Pre-trained teacher model  $\phi$ , condition dataset  $\mathcal{D}$

- 1:  $\theta \leftarrow \text{copyWeights}(\phi)$ ,  $\psi \leftarrow \text{copyWeights}(\phi)$  // intialize
- 2: **repeat**
- 3:   *// Generate tokens  $x_0$*
- 4:   Sample  $x_{\text{init}} \sim p_{\text{init}}$ ,  $c \sim \mathcal{D}$  // with strategy in [Sec. 4.3](#)
- 5:   Get generator logits  $z_\theta(x_{\text{init}}, c) \in \mathbb{R}^{B \times h \times w \times V}$
- 6:    $x_\theta \in \mathbb{R}^{B \times h \times w} \xleftarrow{\text{sample}} p_\theta(x_0 | x_{\text{init}}) = \text{softmax}(z_\theta(x_{\text{init}}, c))$
- 7:   *// Update generator  $\theta$*
- 8:   Sample  $t \sim \mathcal{U}[0, 1]$ ,  $\tilde{x}_t \sim q_{t|0}(\tilde{x}_t | x_\theta(x_{\text{init}}, c))$  // Forward
- 9:   Calculate  $p_\phi(x_0 | \tilde{x}_t, c)$  and  $p_\psi(x_0 | \tilde{x}_t, c)$
- 10:   Update  $\theta$  with the loss gradient  $\nabla_\theta \mathcal{L}_{\text{Di[M]O}}$  (Eq. (7))
- 11:   *// Update auxiliary model  $\psi$*
- 12:   Sample  $t' \sim \mathcal{U}[0, 1]$ ,  $\tilde{x}_{t'} \sim q_{t'|0}(\tilde{x}_{t'} | x_\theta(x_{\text{init}}, c))$
- 13:   Update  $\psi$  with cross entropy loss (Eq. (3))
- 14:   **until** convergence
- 15: **Return** one-step generator  $\theta$

# Content

---

- Preliminaries
- Methods
- Results

# Visual Results



$r_{\text{init}} = 0$

Di[M]O ( $r_{\text{init}} = 1$ ) one step

Di[M]O ( $r_{\text{init}} = 0.6$ ) one step

Teacher 16 steps

# Visual Results

64  
Steps



16  
Steps



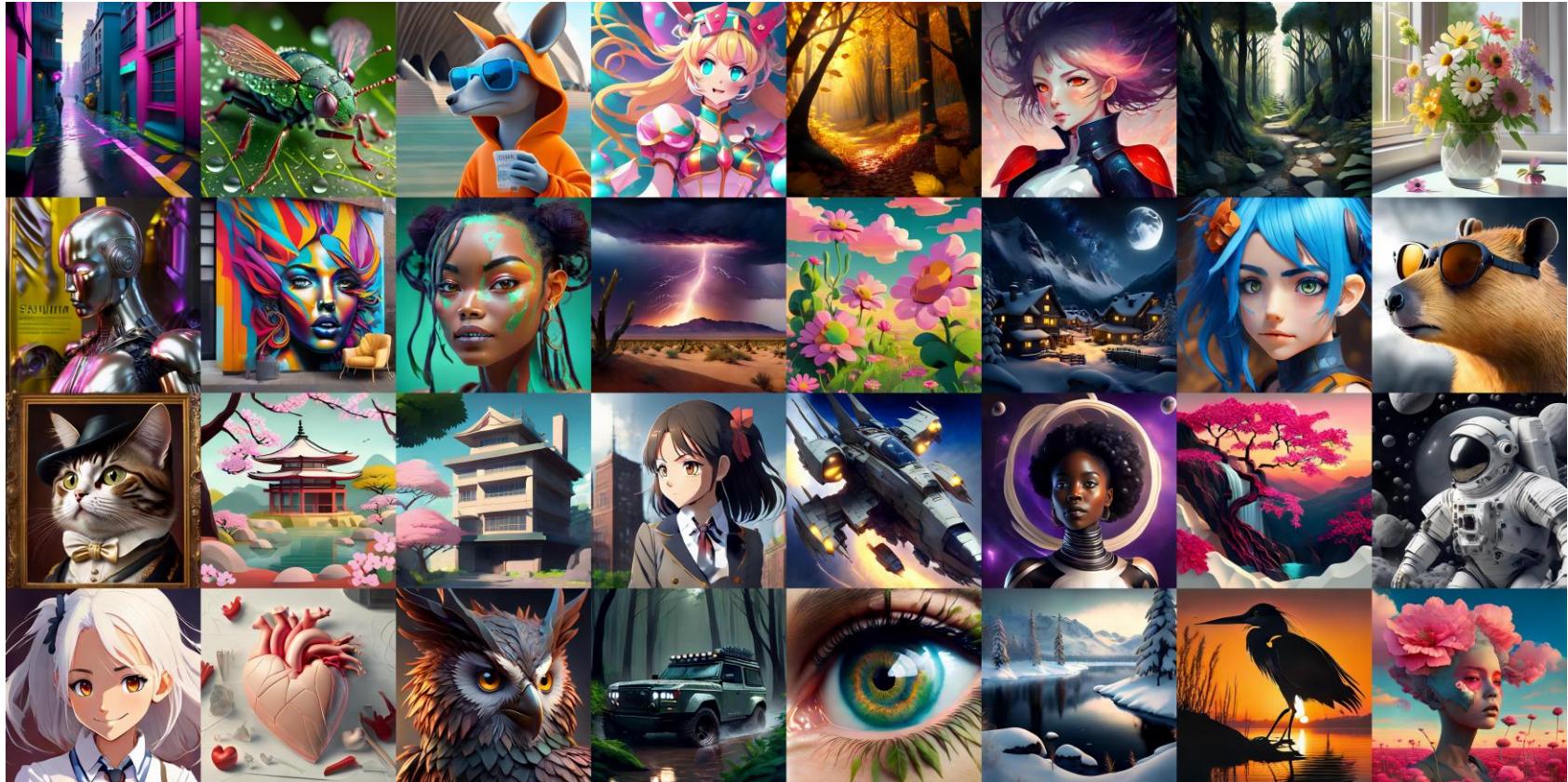
4  
Steps



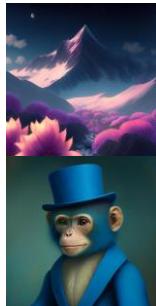
1  
Step  
(ours)



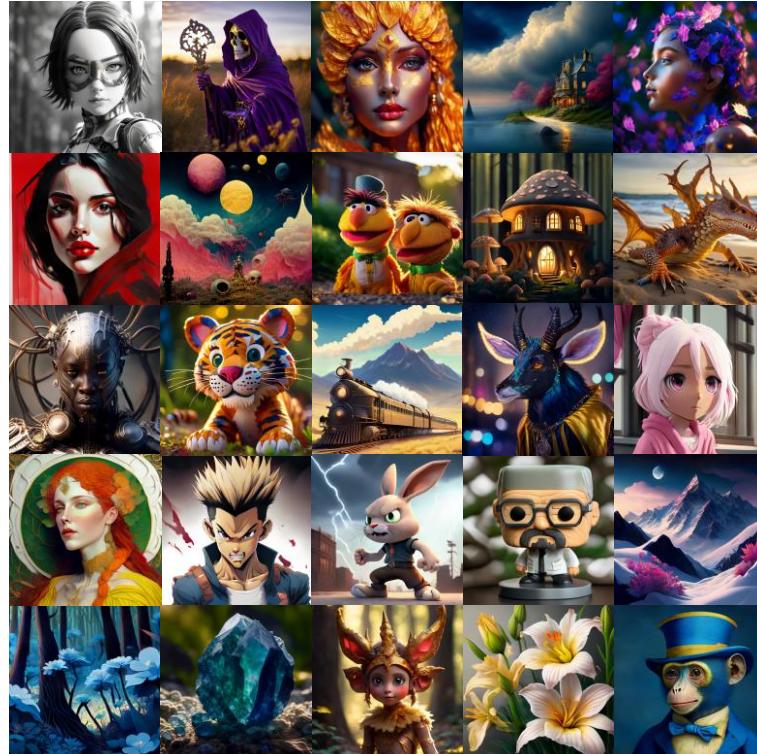
# Visual Results



# Runtime Comparison



10.5 s/image



0.80 s/image

# Quantitative Results (MaskGit Teacher)

ImageNet	Method	Step (↓)	FID (↓)	IS (↑)	Precision (↑)	Recall (↑)	Density (↑)	Coverage (↑)
Teacher	MaskGit [7]	16	6.60	224.07	0.831	0.402	1.246	0.977
	MaskGit [7]	8	6.66	221.57	0.827	0.397	1.233	0.974
	MaskGit [7]	4	10.73	192.29	0.748	0.313	1.011	0.920
	MaskGit [7]	2	91.35	13.37	0.178	0.164	0.091	0.122
Sampler	$\theta$ -trapezoidal [81]*	64	6.7	-	-	-	-	-
	$\theta$ -trapezoidal [81]*	32	7.1	-	-	-	-	-
Distillation	di4c [31]	4	6.79	209.2	-	-	-	-
	di4c-d [31]	4	6.57	213.6	-	-	-	-
	Di[M]O	<b>1</b>	6.91	214.0	0.828	0.377	1.255	0.967

# Quantitative Results (Meissonic Teacher)

HPS v2.0						
Model	Step	Anim.	Concept-art	Painting	Photo	Averaged
Latent Diffusion [83]	25	25.73	25.15	25.25	26.97	25.78
DALL-E 2 [79]	-	27.34	26.54	26.68	27.24	26.95
Stable Diffusion v1.4 [83]	50	27.26	26.61	26.66	27.27	26.95
Stable Diffusion v2.0 [83]	50	27.48	26.89	26.86	27.46	27.17
DeepFloyd-XL [19]	25	27.64	26.83	26.86	27.75	27.27
SDXL Base 1.0 [76]	50	28.88	27.88	27.92	28.31	28.25
SDXL Refiner 1.0 [76]	50	28.93	27.89	27.90	28.38	28.27
InstaFlow [55]	1	25.98	25.79	25.93	26.32	26.01
SD Turbo [90]	1	27.98	27.59	27.16	27.19	27.48
SwiftBrush v2 [18]	1	27.25	27.62	26.86	26.77	27.15
Meissonic [4]	48	<b>29.57</b>	<b>28.58</b>	<b>28.72</b>	<b>28.45</b>	<b>28.83</b>
	32	29.18	28.32	28.28	27.96	28.44
	16	28.61	27.82	27.84	27.32	27.90
	8	25.62	26.49	26.67	27.07	26.46
	4	25.01	24.95	24.87	23.80	24.66
	2	23.06	23.28	23.22	22.38	22.98
	Di[M]O	1	28.64	27.91	27.99	27.92
						28.11

GenEval									
Model	Step	Overall	Objects		Counting		Colors		Color Attribution
			Single	Two			Position		
SD v1.5 [83]	50	0.43	0.97	0.38	0.35	0.76	0.04	0.06	
SD v2.1 [83]	50	0.50	0.98	0.51	0.44	0.85	0.07	0.17	
DALL-E2 [79]	-	0.52	0.94	0.66	<b>0.49</b>	0.77	0.10	0.19	
SDXL [76]	50	<b>0.55</b>	0.98	<b>0.74</b>	0.39	0.85	<b>0.15</b>	<b>0.23</b>	
Meissonic [4]	48	0.54	<b>0.99</b>	0.66	0.42	<b>0.86</b>	0.10	0.22	
	32	0.46	0.92	0.53	0.33	0.80	0.08	0.13	
	16	0.37	0.82	0.39	0.20	0.70	0.05	0.08	
	8	0.20	0.58	0.12	0.05	0.40	0.02	0.04	
	4	0.09	0.31	0.02	0.01	0.18	0.01	0.01	
	2	0.03	0.14	0.01	0.00	0.05	0.00	0.00	
Di[M]O	1	0.43	0.91	0.53	0.22	0.75	0.07	0.11	

Thank You!

# Additional Slides on MDM

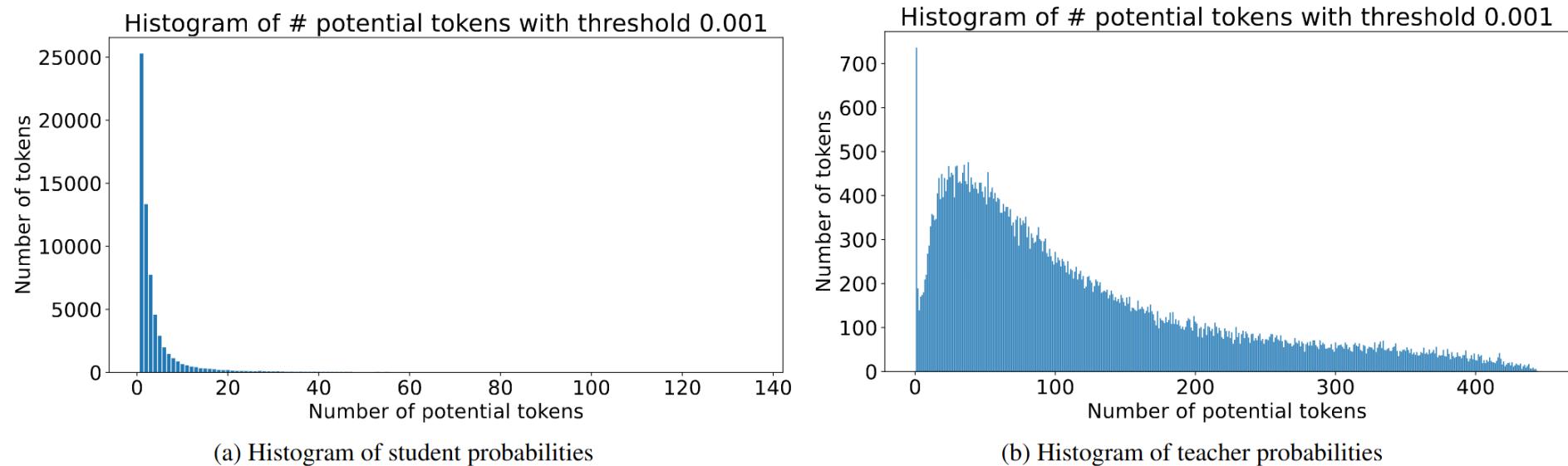


Figure 9. Histogram of number of potential output with probability greater than 0.001

**Entropy shift:** from the output logit to input  $x_{\text{init}}$

# Additional Slides on MDM

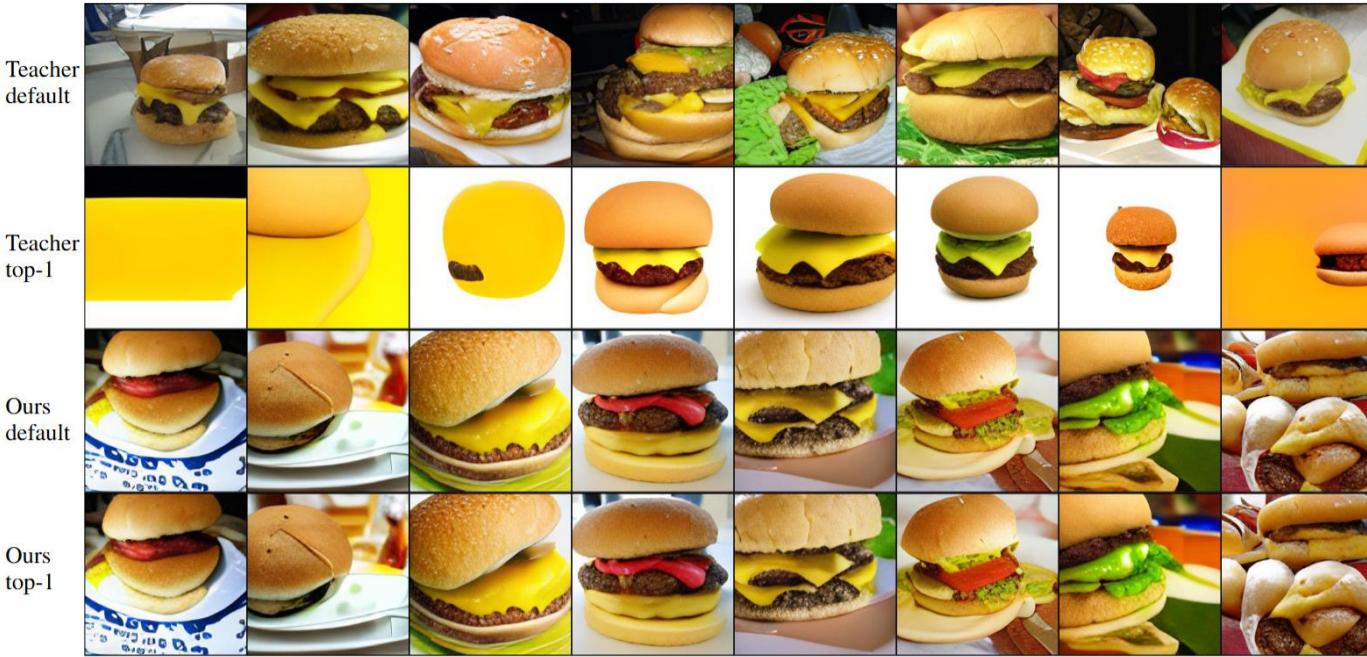


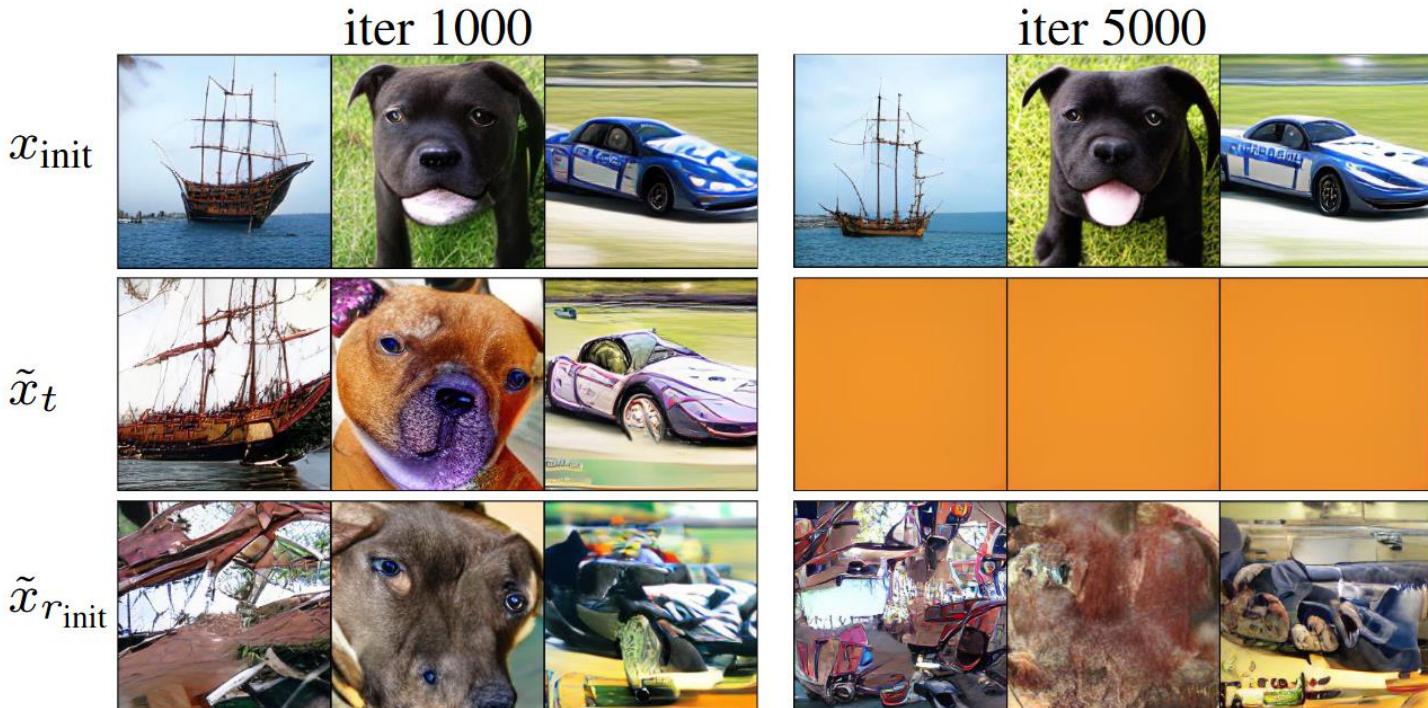
Figure 12. The top two rows display results from the teacher model using 16-step generation with default setup, with the first row generated without top-k filtering and the other with top-1 sampling. The bottom two rows showcase results from our distilled one-step generator. The class label is 933.

**Teacher top-1:** non-typical high likelihood sequence

**Student top-1:** collapsed output logits

# Additional Slides on MDM

Different choices for the input of model  $\theta$  to approximate  $\frac{dz_\theta}{d\theta}$

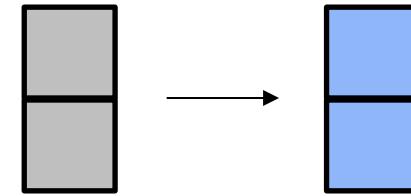


$\tilde{x}_{r_{\text{init}}}$  is constructed by reapplying the initial mask from  $x_{\text{init}}$  to the generated  $x_\theta$

# Additional Slides on Limitations of MDM

- Incorrect joint distribution for multi-token prediction 😞
- Need sufficient steps for better performance 😞

MDM predict two tokens in parallel *at once*  
*each token is sampled independently*



Continuous diffusion predict two pixels in parallel *with multiple-steps*  
*the final output pixels depend on each other*



# Comparison to DMD

---

## Algorithm 1: DMD Training procedure

---

**Input:** Pretrained real diffusion model  $\mu_{\text{real}}$ , paired dataset  $\mathcal{D} = \{z_{\text{ref}}, y_{\text{ref}}\}$

**Output:** Trained generator  $G$ .

```

1 // Initialize generator and fake score estimators
   from pretrained model
2  $G \leftarrow \text{copyWeights}(\mu_{\text{real}})$ ,  $\mu_{\text{fake}} \leftarrow \text{copyWeights}(\mu_{\text{real}})$ 
3 while train do
4     // Generate images
5     Sample  $z \sim \mathcal{N}(0, \mathbf{I})$ ,  $(z_{\text{ref}}, y_{\text{ref}}) \sim \mathcal{D}$ 
6      $x \leftarrow G(z)$ ,  $x_{\text{ref}} \leftarrow G(z_{\text{ref}})$ 
7      $x = \text{concat}(x, x_{\text{ref}})$  if dataset is LAION else  $x$ 
8
9     // Update generator
10     $\mathcal{L}_{\text{KL}} \leftarrow \text{distributionMatchingLoss}(\mu_{\text{real}}, \mu_{\text{fake}}, x)$  // Eq 7
11     $\mathcal{L}_{\text{reg}} \leftarrow \text{LPIPS}(x_{\text{ref}}, y_{\text{ref}})$  // Eq 9
12     $\mathcal{L}_G \leftarrow \mathcal{L}_{\text{KL}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}$ 
13     $G \leftarrow \text{update}(G, \mathcal{L}_G)$ 
14
15     // Update fake score estimation model
16     Sample time step  $t \sim \mathcal{U}(0, 1)$ 
17      $x_t \leftarrow \text{forwardDiffusion}(\text{stopgrad}(x), t)$ 
18      $\mathcal{L}_{\text{denoise}} \leftarrow \text{denoisingLoss}(\mu_{\text{fake}}(x_t, t), \text{stopgrad}(x))$  // Eq 6
19      $\mu_{\text{fake}} \leftarrow \text{update}(\mu_{\text{fake}}, \mathcal{L}_{\text{denoise}})$ 
20 end while

```

---



---

## Algorithm 1 Di[M]O Distillation

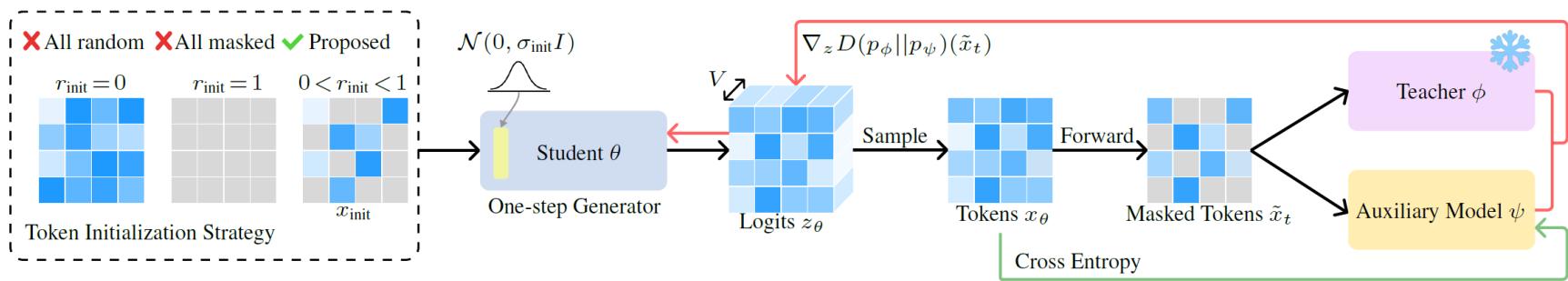
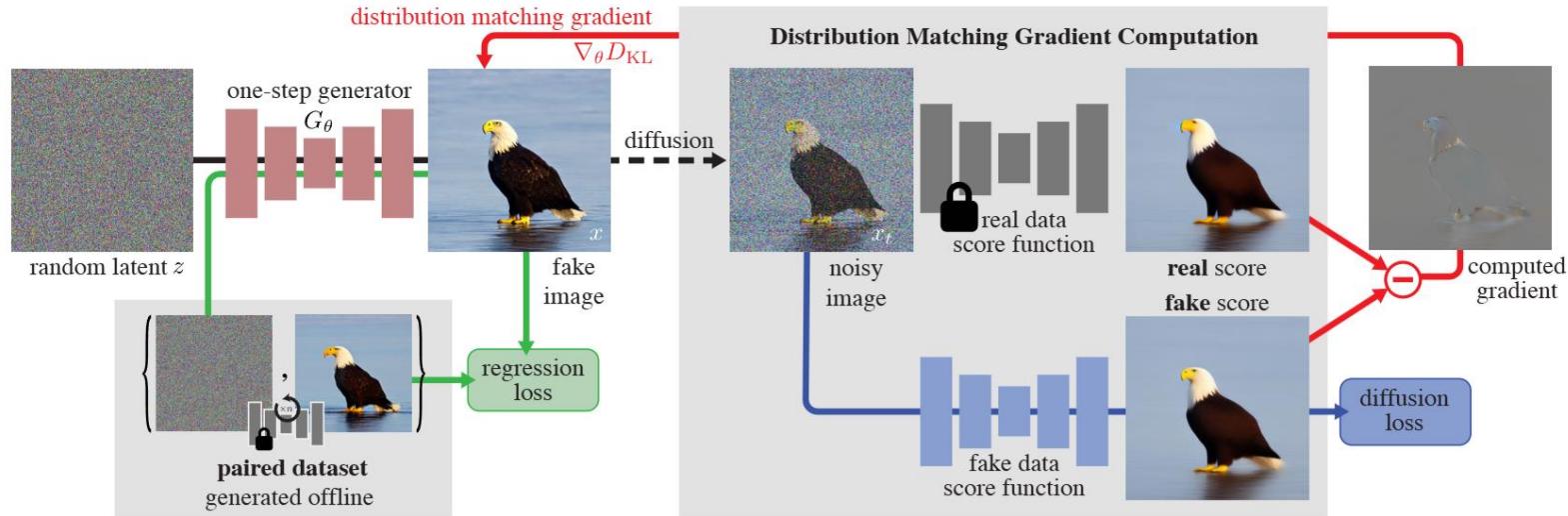
---

**Require:** Pre-trained teacher model  $\phi$ , condition dataset  $\mathcal{D}$

- 1:  $\theta \leftarrow \text{copyWeights}(\phi)$ ,  $\psi \leftarrow \text{copyWeights}(\phi)$  // initialize
- 2: **repeat**
- 3: // Generate tokens  $x_0$
- 4: Sample  $x_{\text{init}} \sim p_{\text{init}}$ ,  $c \sim \mathcal{D}$  // with strategy in [Sec. 4.3](#)
- 5: Get generator logits  $z_{\theta}(x_{\text{init}}, c) \in \mathbb{R}^{B \times h \times w \times V}$
- 6:  $x_{\theta} \in \mathbb{R}^{B \times h \times w} \xleftarrow{\text{sample}} p_{\theta}(x_0 | x_{\text{init}}) = \text{softmax}(z_{\theta}(x_{\text{init}}, c))$
- 7: // Update generator  $\theta$
- 8: Sample  $t \sim \mathcal{U}[0, 1]$ ,  $\tilde{x}_t \sim q_{t|0}(\tilde{x}_t | x_{\theta}(x_{\text{init}}, c))$  // Forward
- 9: Calculate  $p_{\phi}(x_0 | \tilde{x}_t, c)$  and  $p_{\psi}(x_0 | \tilde{x}_t, c)$
- 10: Update  $\theta$  with the loss gradient  $\nabla_{\theta} \mathcal{L}_{\text{Di[M]O}}$  (Eq. (7))
- 11: // Update auxiliary model  $\psi$
- 12: Sample  $t' \sim \mathcal{U}[0, 1]$ ,  $\tilde{x}_{t'} \sim q_{t'|0}(\tilde{x}_{t'} | x_{\theta}(x_{\text{init}}, c))$
- 13: Update  $\psi$  with cross entropy loss (Eq. (3))
- 14: **until** convergence
- 15: **Return** one-step generator  $\theta$

---

# Comparison to DMD



# Comparison to DMD: Key Differences

1. DiMO is tailored for MDM and does not rely on the score function used in continuous diffusion models.
2. While DMD minimize the divergence of *denoising distributions* over the *full space of latent images*, DiMO instead optimize the divergence of *token-level conditional distributions*
3. DiMO generalizes beyond the reverse KL, supporting arbitrary  $f$ -divergences for distribution matching, effectively mitigating the mode-seeking bias

# Future Plan

---

1. Apply to discrete language diffusion models
2. AR Image modeling as special case of MIM?  
(block diffusion)
3. Introduce supervision of real data via GAN  
(require gradient estimation)
4. More rigorous derivation

# Derivations

$$\nabla_{\theta} \mathcal{L}_{\text{Di}[\mathbb{M}]O} \approx \mathbb{E}_{x_{\text{init}}, t} \left[ w(t) \left( \mathbb{E}_{q_{t|0}} \left[ \begin{array}{c} \nabla_{z_{\psi}} D(p_{\phi} || p_{\psi})(\tilde{x}_t) \\ \frac{dz_{\theta}(x_{\text{init}})}{d\theta} \end{array} \right] \right) \right]$$

$$D(p_{\phi} \| p_{\theta})(\tilde{x}_t) := \frac{1}{L_M} \sum_{\substack{i=1 \\ \tilde{x}_t^i = [\mathbb{M}]}}^L D(p_{\phi}(x_0^i | \tilde{x}_t) \| p_{\theta}(x_0^i | \tilde{x}_t))$$

$$\frac{\partial}{\partial z_{\theta_j^i}} p_{\theta}(x_0^i = j | \tilde{x}_t) = \frac{\exp(z_{\theta_j^i})}{\sum_{n=1}^V \exp(z_{\theta_n^i})} - p_{\theta}(x_0^i = j | \tilde{x}_t)^2 = p_{\theta}(x_0^i = j | \tilde{x}_t)(1 - p_{\theta}(x_0^i = j | \tilde{x}_t)),$$

$$\frac{\partial}{\partial z_{\theta_j^i}} p_{\theta}(x_0^i = k | \tilde{x}_t) = -\frac{\exp(z_{\theta_k^i}) \exp(z_{\theta_j^i})}{(\sum_{n=1}^V \exp(z_{\theta_n^i}))^2} = -p_{\theta}(x_0^i = k | \tilde{x}_t)p_{\theta}(x_0^i = j | \tilde{x}_t).$$

# Derivations

$$D_{f_i} = \sum_{k=1}^V p_\theta(x_0^i = k | \tilde{x}_t) f\left(\frac{p_\phi(x_0^i = k | \tilde{x}_t)}{p_\theta(x_0^i = k | \tilde{x}_t)}\right).$$

When the generator function  $f$  is differentiable, we can calculate its gradient as:

$$\begin{aligned} \frac{\partial D_{f_i}}{\partial z_{\theta_j}^i} &= \frac{\partial}{\partial z_{\theta_j}^i} \sum_{k=1}^V p_\phi(x_0^i = k | \tilde{x}_t) f\left(\frac{p_\phi(x_0^i = k | \tilde{x}_t)}{p_\theta(x_0^i = k | \tilde{x}_t)}\right) \\ &= \sum_{k=1}^V \left( \frac{\partial p_\theta(x_0^i = k | \tilde{x}_t)}{\partial z_{\theta_j}^i} \left[ f\left(\frac{p_\phi(x_0^i = k | \tilde{x}_t)}{p_\theta(x_0^i = k | \tilde{x}_t)}\right) - \left(\frac{p_\phi(x_0^i = k | \tilde{x}_t)}{p_\theta(x_0^i = k | \tilde{x}_t)}\right) f'\left(\frac{p_\phi(x_0^i = k | \tilde{x}_t)}{p_\theta(x_0^i = k | \tilde{x}_t)}\right) \right] \right) \end{aligned}$$

$$\nabla_{z_\psi} D_{FKL_i}(p_\phi(x_0^i | \tilde{x}_t) \| p_\psi(x_0^i | \tilde{x}_t)) := \nabla_{z_\psi} D_{FKL_i}(\tilde{x}_t) = (p_\psi(x_0^i | \tilde{x}_t) - p_\phi(x_0^i | \tilde{x}_t)),$$

$$\nabla_{z_\psi} D_{RKL_i}(p_\phi(x_0^i | \tilde{x}_t) \| p_\psi(x_0^i | \tilde{x}_t)) := \nabla_{z_\psi} D_{RKL_i}(\tilde{x}_t) = p_\psi(x_0^i | \tilde{x}_t) \left( \log\left(\frac{p_\psi(x_0^i | \tilde{x}_t)}{p_\phi(x_0^i | \tilde{x}_t)}\right) - D_{RKL_i}(\tilde{x}_t) \right).$$